REMARKS

Applicant thanks Examiner for the detailed review of the application.

Claim Rejections -35 USC § 102(b)

The Office Action has rejected Claims 17-18, 20-28, 32, under 35 U.S.C. § 102(b) as being anticipated by U.S. Pat. No. 5,485,626 to Lawlor et al. (referred to hereinafter as "Lawlor"). However, the Office Action has failed to make a prima facie case of anticipation for the claims, and such rejections should be withdrawn.

"[F]or anticipation under 35 U.S.C. 102, the reference must teach *every aspect* of the claimed invention ..." MPEP 706.02 (emphasis added). "The identical invention must be shown *in as complete detail as contained in the ... claim*." *Richardson v., Suzuki Motor Co*., 868 F. 2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989) (emphasis added).

Applicant's claim 17, as amended, includes the element, "a shared register to be addressable by a user-level instruction, the shared register to be **directly accessible by at least the first shred and the second shred** to provide communication between the first shred and the second shred," (emphasis added). In contrast, Lawlor discloses the following in column 17:

> An SRQ is an object used to exchange information between threads and to synchronize the transfer of control between threads. One thread can communicate with another thread by issuing a send type instruction to an SRQ or an SRC. Another thread can then obtain the information from the queue or counter by issuing a receive type instruction. These facilities are useful as a basis for many forms of efficient inter-thread communication and synchronization. Thread synchronization is provided by using Thread Send/ Receive Messages and an SRQ in the following manner.

As can be seen, Lawlor describes an **object** to exchange information between threads, i.e. SRQ, which is a queue or counter. One active thread issues an instruction, such as a receive or send message/instruction, to retrieve or place information from/on the queue. Lawlor does describe in col. 19 lines 31-32, "in essence, this allows for register to register communications between

threads." Note that Lawlor explicitly requires communication from one register to another, i.e. between two registers, instead of allowing a shared register to be "directly accessible," as stated in applicant's claim 17. Furthermore, note the term "in essence" from Lawlor's disclosure illustrates that the disclosed process is not actually register to register communication, as a send message places information on the queue from a first register and a receive instruction retrieves the information from the SRQ queue, but rather Lawlor describes register to queue to register communication.

In contrast, applicant's claim 17 states that the shared register is "directly accessible … to provide communication between the first and second thread." Examples and embodiments of local inter-shred communication is discussed in aplicant's detailed description at parapgraph 0072, "Local data synchronization is performed utilizing atomic register updates, register semaphores, or memory semaphores." For example, a single shared register may be updated by a first shred to update a data value/location directly accessible by the first and a second shred. In contrast, Lawlor would require a send message to place the new value from one thread's register on a queue and the then a receive message to retrieve the value from the queue and place it in another thread's register. Lawlor disclosure does not suggest sharing of register, but rather communication between threads through a object, such as a queue. Therefore, applicant respectfully submits that claim 17, as well as its dependent claims 20-28 and 32-34, are in condition for allowance for at least the reasons stated above.

Claim Rejections -35 USC § 103(a)

The Office Action has rejected Claims 1-12, 15-16, 35-44, 45-46, 50, and 62 under 35 U.S.C. § 103(a) as being unpatentable over Lawlor in view of Galvin et al. (herein referred to as "Galvin").

"The examiner bears the initial burden of factually supporting any prima facie conclusion of obviousness." MPEP § 2142. It is well established that *prima facie* obviousness is only established when three basic criteria are met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991) (MPEP 2144). The Office Action has failed to meet one or more of these requirements.

Applicant's claim 1, as amended, includes the element, "creating, responsive to the programming instruction, a first shared resource thread (shred) that **is associated with a first private portion of a first application state** and **shares a second shared portion of the first application state with at least a second shred**, wherein creating the first shred is performed without the intervention of an operating system," (emphasis added). Lawlor discloses at col. 10 lines 65 – col. 11 lines 5 the following:

> In FIG. **1**, the state vector (SV) register **110** holds the operating and control state of the processors $\mathbf{102}_1$, $\mathbf{102}_2$ and $\mathbf{102}_3$. From this state, the context of the program is determined by the token processor **1005**, shown in more detail in FIG. **3**. Similarly, the SV register **112** holds the operating and control state of processor **106** from which the context of the program is determined by the token processor **1007**. It

As can be seen, Lawlor expressly requires that SV register 110 holds the operating and control state

of processor 102 (1-3) and SV register 112 holds the states for processor 106. Therefore, as can be seen, there is no "first private portion" and "second shared portion" of an application state, as described in applicant's claim 1. As described in reference to Lawlor's **Figure 3**, address and authority information in held in SV 110, is essential to determining Lawlor's authority to access objects. Furthermore, as described above, communication between threads is done utilizing an object, such as a queue. Therefore, combination of Lawlor with a reference teaching use of a private portion and a shared portion, such as in applicant's claim 1 would be impermissible, as an essential element of Lawlor's disclosure would be negated by the combination.

However, even if combination with Galvin would be permitted, Galvin also does not disclose portions of an application state. Galvin states that a thread sometimes consists of a program counter, a register set, and a stack space (See section 4.5.1). Yet, he does not suggest that any part/portion of the register set is private, such as replicated for each shred, and another part/portion the register set is shared among other shreds. Galvin states, "Threads can create child threads, and can block waiting for system calls to complete, if one thread is blocked, another thread can run." Note that Galvin's discussion on page 112 focuses on sequential execution of threads, not concurrent execution of threads. Therefore, only a single program counter and register set may be provided to switch the thread and child threads in and out, when one of the threads block. In contrast, applicant's claim 1 deals with executing concurrently a first shred and a second shred, and also includes a private portion of the first application state associated with the first shred and a shared portion of the application state shared between the first and the second shreds. Galvin only discusses access to all addresses/stacks in a task by all the threads, i.e. the memory state/space, but does not discuss sharing of the application state or private portions of the application state. One embodiment discussed in applicant's detailed description includes use of per-shred registers to hold per shred application state, but neither Galvin or Lawlor private/shared portions of register sets

and/or application state.

Applicant's claim 35 includes the element, "executing the plurality of threads of execution **concurrently on multiple instruction sequencers of a processor in the machine**," (emphasis added). As stated above, Galvin only discusses sequential execution of threads sharing a CPU, not "execution concurrently." Furthermore, Lawlor at col. 12 lines 31-32 discloses the following:

on the TDQ in priority sequence. There are N levels of priority. Each task has a priority key which can be changed under program control. The active task causes instructions to be executed whereby work is performed, or it communicates with other tasks requesting the other tasks to do some work. The other tasks are either in an inactive dispatchable or inactive waiting state. The instruction fetch cycle of the

As can be seen, Lawlor explicitly states that other tasks are in an inactive state. Only the active task causes instructions to be executed, not the inactive task. Furtheremore, Lawlor's system illustrated in Figure's 1 and 2 include multiple processor with single thread execution on each processor at a time, not concurrent execution on multiple instructions sequencers **of a processor**. In applicant's claim 35 two active shreds may cause instructions to be executed on multiple instruction sequencers of a single processor concurrently. Furthermore, neither Lawlor or Galvin mention the use of instruction sequencers. Consequently, applicant respectfully submits that independent claims 1 and 35, as well as their dependent claims, are now in condition for allowance for at least the reasons stated above.

The Office Action has rejected Claims 51, 55-58, 63-66 under 35 U.S.C. § 103(a) as being unpatentable over Lawlor in view of Patterson et al. (herein referred to as "Patterson"). Applicant's claim 51 includes the element, "a microprocessor implementing an instruction set architecture (ISA), the microprocessor capable of executing multiple OS-generated threads, wherein the microprocessor is also capable of concurrently executing multiple shared resource threads (shreds) associated with one of the OS-generated threads." Lawlor does no disclose executing "OS-generated threads" and does not disclose "**concurrently executing** multiple shreds that are

associated with one of the OS-generated threads." Patterson's disclosure of making a superscalar processor with instructions that can run in parallel also does not teach concurrently executing multiple user-level generated shreds that are associated with one of the OS-generated threads. In fact, Patterson only suggests use of parallel instructions, not association of shreds with an OS generated thread or concurrent execution of such. Similarly, applicant's claim 55 includes the element, "the microprocessor is further to execute the user-level threads concurrently." Therefore, for at least the reasons stated above, the applicant respectfully requests that claims 51 and 55, as well as claim 55's dependent claims, are now in condition for allowance.

Claim 67 has been amended, and now includes, "a processor capable of user-level multithreading including: a first group of resources to hold a per shared resource thread ("shred") application state for a first shred … a second group of resources to hold a per shred application state for a second shred … third group of resources to be shared by the first shred and the second shred to hold a shared application state…and execution resources to concurrently execute the first shred and the second shred." None of Lawlor, Galvin, or Patterson describe a processor including resources to hold private or shared applications states, as well as execution resources to execute shreds concurrently. Therefore, applicant respectfully requests that claim 67, as well as its amended dependent claim 68 and newly added dependent claims 69-71, are in condition for allowance.

If there are any additional charges, please charge Deposit Account No. 50-0221. If a telephone interview would in any way expedite the prosecution of the present application, the Examiner is invited to contact David P. McAbee at (503) 712-4988.

Respectfully submitted,
Intel Corporation

Dated: June 5, 2007

/David P. McAbee/Reg. No. 58,104/
David P. McAbee
Reg. No. 58,104

Intel Corporation
M/S JF3-147
2111 NE 25th Avenue
Hillsboro, OR 97124
Tele – 503-712-4988
Fax – 503-264-1729